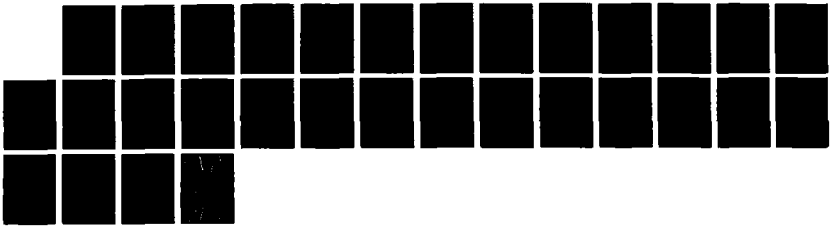
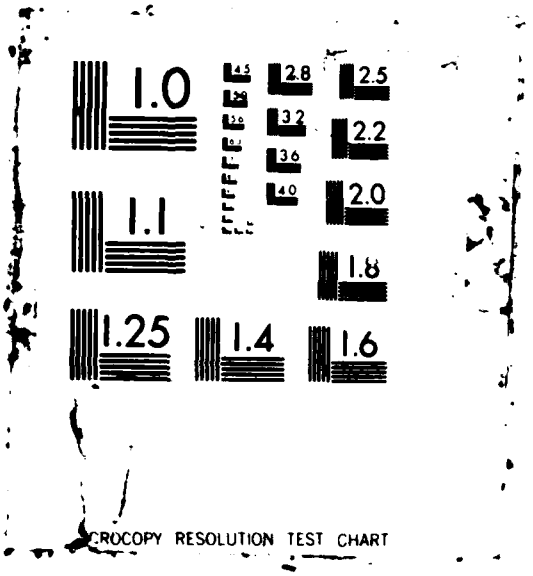
AD-A179 149 MODELING APPROACHES AND SYSTEMS RELATED TO STRUCTURED MODELING(U) CALIFORNIA UNIV LOS ANGELES WESTERN MANAGEMENT SCIENCE INST A M GEOFFRION FEB 87 1/1

UNCLASSIFIED WMSI-WP-339 N00014-75-C-0570 F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
CROCOPY RESOLUTION TEST CHART

Working Paper No. 339

MODELING APPROACHES AND SYSTEMS RELATED TO STRUCTURED MODELING

by

ARTHUR M. GEOFFRION

Contract N00014-75-C-0570

February 1987

WESTERN MANAGEMENT SCIENCE INSTITUTE
University of California, Los Angeles

WESTERN MANAGEMENT SCIENCE INSTITUTE
University of California, Los Angeles

Working Paper No. 339

Drafted July, 1986
Revised February, 1987

MODELING APPROACHES AND SYSTEMS RELATED TO STRUCTURED MODELING

by

Arthur M. Geoffrion

Abstract

This companion to the author's paper "An Introduction to Structured Modeling" discusses modeling approaches by other authors and some of the modeling systems available as alternatives for at least some of the functions to which a structured modeling system aspires. It is an expansion of the brief discussion of this topic in the mentioned paper, with which the reader is assumed to be familiar.

Partially supported by the National Science Foundation, the Office of Naval Research, and the Navy Personnel R&D Center. The views contained in this report are those of the author and not of the sponsoring agencies.

MODELING APPROACHES AND SYSTEMS RELATED TO STRUCTURED MODELING

Many modeling approaches and systems (or classes of systems) share at least some of the aspirations of structured modeling. This paper discusses many of the most pertinent of these in the context of the eight desirable modeling system features proposed in Section 1.2 of Geoffrion <1986a>. Those features are repeated here for the reader's convenience.

(a) a rigorous and coherent conceptual framework for modeling based on a single model representation format suitable for managerial communication, mathematical use, and direct computer execution

(b) independence of model representation and model solution, with model interface standards to facilitate building a library of models and of easily accessed solvers for retrieval, systems of simultaneous equations, optimization, and other important manipulations

(c) sufficient generality to encompass most of the great modeling paradigms that MS/OR and kindred model-based fields have developed for organizing the complexity of reality (activity analysis, decision trees, flow networks, graphs, markov chains, queueing systems, and so on)

(d) usefulness for most phases of the entire life-cycle associated with model-based work

(e) representational independence of general model structure and the detailed data needed to describe specific model instances

(f) desktop implementation with a modern user interface (e.g., visually interactive, directly manipulative, syntactically humane, and with liberal use of graphics and tables)

(g) integrated facilities for data management and ad hoc query in the tradition of database systems

(h) immediate expression evaluation in the tradition of desktop spreadsheet software.

The categories of discussion are:

1. Relational database systems
2. GAMS
3. PLATOFORM
4. Other modern mathematical programming systems
5. IFPS and OPTIMUM
6. ANALYTICOL
7. Conceptual graphs
8. Discrete event simulation
9. Graph grammar-based systems
10. Special purpose systems.

Three general conclusions can be drawn from the discussion that follows and from the evolution of structured modeling over the last several years. First, structured modeling has received significant inspiration and reinforcement from some of these approaches and systems. Second, there remain many attractive opportunities for structured modeling to cross-fertilize with other approaches. Third, none of the other approaches or systems reviewed comes close to achieving all eight of the features. This is not meant as a criticism, as many of the systems are superb for their intended uses, but rather as a way of distinguishing the niche that structured modeling attempts to fill.

Software packages are referenced either by an authored publication in the usual way or by listing their name in italics, in which case their vendor is noted in a separate reference section appearing at the end of this paper.

1. Relational Database Systems

Consideration is limited here to relational database management systems available for desktop computers. There is an ample supply of such systems; a recent survey (Krasnoff and Dickinson <1986>) lists 56 such. Highly rated products include *DataEase, dBASE III Plus, Paradox, R:base 5000*, and *Revelation*.

Feature (a). Ideally, each of these systems offers a coherent conceptual modeling framework, namely the relational data model (e.g., Tsichritzis and Lochovsky <1982>). Most systems actually offer one or another variant of the standard relational

model. Often the exact variant offered is not stated explicitly and must be inferred from the documentation or the program itself.

Feature (b). Only one solver is involved, namely the query processor, and so there is no provision for a "library" of solvers or for interface standards through which other solvers can be invoked.

Feature (c). The basic data model of these systems is not general enough to encompass the great modeling paradigms of MS/OR. Omission to provide for mathematical functions is one of the fatal limitations.

Feature (d). These systems do a fairly good job of life-cycle support relative to typical database applications. However, the life-cycle of analytical modeling applications tends to be more complex.

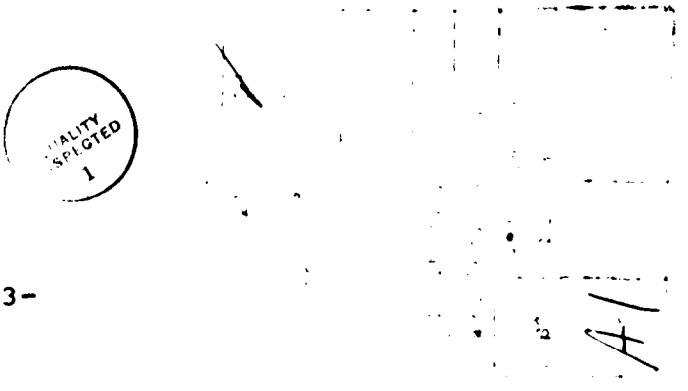
Feature (e). The general model structure is usually evident in the structure of the tables that hold the data. So evident, in fact, that some desktop systems neglect to declare it separately and thus violate the spirit of this feature -- in contrast to mainframe database systems, nearly all of which exhibit this feature to a high degree.

Feature (f). The user interfaces of these desktop packages are often excellent.

Feature (g). The database capabilities of these packages are, of course, their main strength, but they are not integrated with any significant facilities useful for analytical modeling. However, at least one package -- *KnowledgeMan* -- does integrate relational database capabilities with spreadsheet facilities useful for some kinds of analytical modeling.

Feature (h). Those packages which allow "calculated" or "virtual" fields have a kind of immediate expression evaluation capability.

Database systems have reached a high degree of sophistication. However, their modeling scope is not designed to serve the needs of the analytical modeler. Cross-breeding these systems with analytical modeling systems could well result in a qualitative improvement in their usefulness. Recent efforts in the extensible database systems area may well make this practical (e.g., Batory and Mannino <1986>).



2. GAMS

GAMS -- for General Algebraic Modeling System -- has been in use at the World Bank for a number of years for optimization-based economic development studies (Bisschop and Meeraus <1982>, Kendrick and Meeraus <1985>).

Feature (a). GAMS adopts the mathematical programming paradigm as its conceptual framework. It uses conventional mathematics (mainly sets and algebra) as the basis of its model representation. This representation is computer-executable and fine also for mathematical use, but of course is of limited value for managerial communication.

Feature (b). GAMS succeeds admirably in making model representation and model solution independent of one another, and has a well interfaced library of solvers for nonlinear as well as linear programming. Construction of the optimizer input file (matrix generation) is entirely automatic and transparent to the user.

Feature (c). GAMS is not intended to support anything other than the mathematical programming paradigm.

Feature (d). GAMS is primarily intended to support just the optimization phase of the modeling life-cycle.

Feature (e). GAMS discourages the separation of general model structure from detailed data; usually the two are intertwined.

Feature (f). GAMS has been rewritten recently in PASCAL for personal computers. The command-driven user interface is not modern.

Feature (g). The PC version of GAMS has limited database capabilities, although it does store its data in a relational database.

Feature (h). GAMS does not provide for immediate expression evaluation.

GAMS is an excellent productivity tool for mathematically oriented users who need to solve mathematical programming problems.

Comparisons of GAMS and structured modeling are available in Geoffrion <1986b> and Kendrick and Krishnan <1986>.

3. PLATOFORM

PLATOFORM -- PLanning TOol written in *DATAFORM* -- is a remarkable long term success story from Exxon Corporation that has only recently been revealed (Palmer <1984>). Its purpose is to support the formulation, solution, analysis, and maintenance of mathematical programming models using an approach that differs radically from standard matrix generation technology. This it apparently does extremely well. To quote from Palmer <1984> (page 20): "... PLATOFORM creates an environment, rather than delivering an application package ... This broad set of capabilities has improved analyst/user productivity tenfold ... In about a month, an entirely new application can be developed that otherwise would have taken over a year ... [there is a] tenfold reduction in elapsed time required to develop a model or to obtain a valid case study." It is used today for essentially all Exxon LP models worldwide.

Feature (a). PLATOFORM adopts the mathematical programming paradigm and also a table-oriented data model for database design. The representation is not well suited to mathematical use and probably not intended for managerial communication.

Feature (b). One could wish for a greater degree of independence between the model and the solver. In structured modeling terms, nearly everything that corresponds to entity or attribute genera is set down cleanly in tables, but much of what corresponds to function and test genera is embedded in the generalized data-driven matrix generator and the control tables used to invoke it. A very elaborate optimization system is coupled with the modeling facilities. In addition to large scale linear programming, it also has facilities for integer, nonlinear, and sequential linear optimization.

Feature (c). PLATOFORM is wedded to the two modeling paradigms mentioned under feature (a).

Feature (d). PLATOFORM provides extensive facilities for data management (which, incidentally, GAMS does not), matrix generation, optimization, and report-writing. It does not provide much support for solution analysis.

Feature (e). PLATOFORM achieves a high degree of independence between general model structure and detailed data.

Feature (f). PLATOFORM was strictly a mainframe system at the time Palmer <1984> was written. However, PLATOFORM takes great care to relieve the user of having to know JCL, PCL, or any other procedural programming language.

Feature (g). Although PLATOFORM has extensive data management facilities, it does not have facilities for general ad hoc queries.

Feature (h). PLATOFORM does not support immediate expression evaluation.

There is much to admire about PLATOFORM. However, its historical evolution from early if venerable LP modeling software leaves it with some serious liabilities as an inspiration for future modeling system designs.

4. Other Modern Mathematical Programming Systems

GAMS and PLATOFORM are not the only mathematical programming systems to appear in recent years that lighten greatly the burden on the user by comparison with the standard commercial offerings.

Nearly all such systems adopt the mathematical programming paradigm as their conceptual framework and thus are not very compatible with the other great modeling paradigms. Most are oriented mainly toward quantitatively skilled people who can use a computer but cannot (or do not wish to) program it. They all offer independence of model representation and model solution, but differ widely in the degree of independence between general model structure and detailed data. Usually they offer just one solver, for which the traditionally onerous matrix generation step is handled entirely by the system, and have no provision for an entire library of solvers. Typically they focus primarily on the optimization phase of the modeling life-cycle. Some are available for desktop machines, and those which are not probably could be adapted to the desktop environment. Only a few have integrated database query capabilities or facilities for immediate expression evaluation.

Like GAMS and PLATOFORM, these systems boost productivity for mathematical programming applications and incorporate some advances worthy of imitation by future modeling systems. However, I believe that they lack the scope and integrated capabilities necessary to revolutionize modeling on a broad scale.

Some of these systems are now discussed briefly. Others are AMPL by Fourer, Gay and Kernighan <1987>, EMP (Expert System for Mathematical Programming) by Schittkowski <1985>, GXMP (Generalized Experimental Mathematical Programming System) by Dolk <1986>, LPMODEL by Katz, Risman and Rodeh <1980>, *MIMI/LP* (Manager for Interactive Modeling Interfaces/Linear Programming), and *PAM* (Practitioner's Approach to Modeling). See Fourer<1983> for a particularly informative discussion of modeling languages for mathematical programming, and Waren, Hung and Lasdon <1986> and Maturana <1987> for surveys of several modern systems.

CAMPS (Lucas and Mitra <1985>) -- Computer Assisted Mathematical Programming System -- is a descendent of UIMPS (Ellison and Mitra <1982>). Its main distinction is that it is almost entirely menu-driven. Presently it runs only on a mainframe. To enhance its capabilities for solution analysis, some work has been done to integrate CAMPS with ANALYZE (Greenberg <1983>).

LINDO -- Linear, Interactive aNd Discrete Optimizer -- is widely used, particularly in academe. It features interactive problem input using fully explicit conventional but indexless mathematical notation. This is handy for small problems, but it violates the principle of separating general model structure and data and is onerous for large problems (for which the user is advised to code a custom matrix generator). *LINDO* runs on most desktop machines and minicomputers.

What's Best! marries *LINDO* with the best selling spreadsheet program, *1-2-3*, which is used for all model-building and solution reporting chores. The marriage is one of truly remarkable simplicity; *LINDO* is all but invisible to the user. Adopting *1-2-3* as the modeling environment confers many advantages, including (obviously) understandability by many managers and immediate expression evaluation, but carries with it a big liability: lack of independence between general model structure and detailed data.

MLD (Burger <1981>, Burger <1982>) -- Modeling Language and Database -- adopts a broader perspective and is closer in spirit to structured modeling than most of the other entries in this category. For example, it consciously exploits the close relationship between modeling and data base design and programming language design; it champions the separation of models and solvers, and also the independence of general model structure and detailed data; it appreciates the virtues of modularity; and it seeks to achieve an interactive system with relational database query facilities integrated with optimization capability. Like GAMS, MLD specifies model structure algebraically.

5. IFPS and OPTIMUM

IFPS is the best selling mainframe financial planning language. *OPTIMUM* is a version of *IFPS* that incorporates linear, nonlinear and integer programming capability (Roy, Lasdon and Lordeman <1986>).

Feature (a). The underlying conceptual framework is similar to that of spreadsheet software like *1-2-3* in that it is based on one- or two-dimensional arrays of text entries, numbers, or algebraic formulas. However, it has a stronger bias toward arrays of identical horizontal dimension -- each horizontal unit typically corresponds to a time period -- and it has built-in facilities for Monte Carlo simulation. Also, its representation

is more like a declarative programming language and is less visually oriented or directly manipulative in character. Only one model representation is used. It is well suited to managerial communication and direct computer execution, but less so to mathematical use (e.g., there is no explicit indexing).

Feature (b). Model representation and solution are reasonably independent, but there is no support at present for the concept of an open solver library.

Feature (c). *IFPS* was designed for just one model paradigm, namely the standard spreadsheet view of financial planning. It turns out that this paradigm is flexible enough to accommodate numerous other paradigms (e.g., Bodily <1986>, Plane <1986>), although this does not always lead to natural representations.

Feature (d). The *IFPS* family of products endeavors to support most of the modeling life-cycle.

Feature (e). *IFPS* normally interweaves general model structure with detailed data. However, "datafile" options do support separation.

Feature (f). Desktop versions of *IFPS* are available as *IFPS/Personal* and *MindSight*, but optimization capability is only available on the mainframe. The user interface for invoking optimization is extremely easy to use, including completely transparent matrix generation.

Feature (g). *IFPS* has modest data management capabilities and essentially no ad hoc query capability.

Feature (h). *IFPS* has reasonably immediate (but not automatic) facilities for expression evaluation.

IFPS and *OPTIMUM* provide a productive modeling environment for applications that fit into the spreadsheet paradigm.

6. ANALYTICOL

ANALYTICOL is a UNIX-based environment for analytical computing in use at AT&T Bell Laboratories (Childs and Meacham <1985>; see also Childs and Vokolos <1986>). Comprising a collection of loosely integrated tools for data extraction, large file manipulation, algebraic modeling, data management, application development, and other functions, it is credited with increasing the productivity of business analysts and application developers by a factor of five or more.

Feature (a). The philosophy of ANALYTICOL is not to impose any particular conceptual framework on users, but rather to support whatever users want to do as well as possible. Of course, some of the tools within ANALYTICOL must be based on

conceptual frameworks for data modeling or analytical modeling. The most interesting of these is the graphical framework of HEQS, which stands for Hierarchical Equation Solver (Derman and Sheppard <1985>). It is based on a directed graph whose nodes are model variables and whose arcs represent computational dependencies among the variables. This is quite similar to structured modeling's element graph.

Feature (b). ANALYTICOL pays no particular attention to the separation of models and solvers.

Feature (c). ANALYTICOL aims to be a capable host for a great many modeling paradigms, although its tools may support some of them better than others.

Feature (d). ANALYTICOL supports all five of the generic activities that its designers believe are associated with analytical studies. These are data acquisition, data refinement and structuring, data analysis and model building, report and graph generation, and specialized tool building.

Feature (e). ANALYTICOL pays no particular attention to the separation of general model structure and detailed data, although it provides some tools for the user who is concerned about this.

Feature (f). ANALYTICOL targets minicomputers and mainframes but does run on suitable UNIX workstations for small applications.

Feature (g). ANALYTICOL provides loosely integrated database capabilities.

Feature (h). ANALYTICOL does not support immediate expression evaluation.

This an impressive system. It represents a polar alternative to the direction taken by structured modeling: instead of providing a unifying conceptual framework and a comprehensive arsenal of supporting tools that are tightly integrated with that framework, ANALYTICOL provides a collection of loosely integrated tools that are of general utility for any conceptual framework.

7. Conceptual Graphs

The most fundamental problem in modeling may well be how to represent knowledge. The MS/OR community generally has taken the attitude that the available spectrum of conventional mathematical notation suffices for this purpose. The artificial intelligence community, on the other hand, has been much more concerned with devising new knowledge representation schemes. Various approaches to knowledge representation can be distinguished; one

standard reference (Barr and Feigenbaum <1981>) lists logic, procedural representations, semantic networks, production systems, analogical representations, semantic primitives, and frames and scripts. One of the most popular of these is semantic networks, which has essentially merged with frames and scripts in recent years (Brachman and Levesque <1985>). This is the viewpoint from which Sowa <1984> has recently brought to culmination his long term effort to unify the foundations of artificial intelligence in terms of "conceptual graphs".

Sowa's theory of conceptual graphs bears a striking similarity to structured modeling that is sketched in Appendix 1. The following general comments are pertinent to both that discussion and the present one.

- Sowa's roots are in the cognitive sciences, especially linguistics, philosophy, and psychology, rather than in the decision sciences.

- Sowa's conceptual graphs are broader in scope than most prior "semantic network" representations used in AI (see Brachman <1979> for a detailed survey). For example, he shows that they offer "a complete notation for first-order logic with direct extensions to modal and higher-order logic" (p. 23), and he explains connections to several other knowledge representation approaches.

- Sowa explains how to map conceptual graphs to and from natural language.

- Conceptual graphs stress semantics rather than syntax. Each graph represents a single declarative proposition, which can be highly complex. A linear notation suitable for typing is available as an alternative to the diagrammatic one.

Conceptual graphs are closely related to genus and element graphs in structured modeling. They have two kinds of nodes: (a) *concept* nodes, which usually represent entities, events, and states, and (b) *conceptual relation* nodes, which represent the roles that concept nodes play in relation to one another. Conceptual graphs are finite, connected, directed, bipartite (every arc links a concept node with a conceptual relation node), and typed in that each node is labeled according to the *type* of the concept or conceptual relation to which it belongs. All conceptual relation nodes have out-degree one. Nearly all conceptual relation nodes have in-degree one; if the in-degree is greater than one, then the incoming arcs bear sequence numbers to distinguish them.

Consider now the eight desirable features listed at the outset. Keep in mind that Sowa's work is theoretical in character; his book does not treat implementation issues to any

significant extent. Thus it may not be totally fair to comment on Sowa's work in terms that presume at least a functional description of a modeling system.

Feature (a). Sowa's development offers a reasonably rigorous and coherent framework for modeling with a single focal representation. He demonstrates that this representation is suitable for mathematical use, at least for applications of first-order logic. He also indicates how this representation can be mapped to and from natural language, an exciting possibility that certainly carries with it the opportunity of good managerial communication (although, strictly speaking, this communication involves a change of model representation). He does not claim direct computer executability of conceptual graphs, but does cite some partial implementations by others (pp. 324-5). In addition, seven implementation projects are cited in Sowa and Way <1986>.

Feature (b). Sowa's development makes a clear distinction between model representation and model solution.

Feature (c). Conceptual graphs were not designed for analytical modeling, but rather for modeling natural language in the context of artificial intelligence. Any MS/OR model can be expressed in natural language, and hence probably also as a conceptual graph, but it does not necessarily follow that conceptual graphs encompass the great modeling paradigms of MS/OR in a useful way. One serious limitation, for example, is the lack of any direct way to model mathematical functions.

Feature (d). Again, one would have to say that Sowa's development was not intended to support work based on analytical modeling or databases.

Feature (e). Unfortunately, although Sowa makes a strong distinction between general structure and detailed data, conceptual graphs merge the two. Sowa implicitly acknowledges (Sec. 6.4) that this leads to inefficiency in database applications, but points out that this is done commonly in knowledge representations of artificial intelligence because AI typically has a low ratio of data items to data descriptors. Database applications, of course, have a very high ratio, and analytical modeling falls somewhere in between.

Feature (f). No mention is made of any desktop implementations among those cited in Sowa and Way <1986>.

Feature (g). Conceptual graphs provide a theoretical foundation for ad hoc query in an even broader sense than is customary in the database field.

Feature (h). Sowa showed how "actors" capable of calculation can be attached to conceptual graphs. It therefore seems possible that an implementation could be designed to support expression evaluation in a reasonably immediate way.

We remind the reader that Appendix 1 contains a fairly detailed discussion of the relationship between conceptual graphs and structured modeling.

8. Discrete Event Simulation

The technology of discrete event simulation developed historically quite independently of the technology of analytical modeling. This is partly a reflection of the differing aims of the two fields, and partly a consequence of the relative absence of mathematics in the heritage of simulation. The main concern of discrete event simulation is mimicking the time-dependent behavior of some target system.

Structured modeling, by contrast, is mainly concerned with representing the pertinent essence of the system itself, and prefers to regard generating the time-dependent behavior as a non-modeling task best left to a solver. (However, it is sometimes possible to design a structured model so that the evaluation operation yields the desired time-dependent behavior.) For this reason, and also because discrete event simulation does not pretend to cope with any of the other great modeling paradigms, it may seem justified to dismiss it from further consideration as an alternative to structured modeling. Nevertheless, a number of developments in simulation are of interest from the structured modeling viewpoint.

One particularly interesting development is the emergence, especially during the last decade, of a heightened awareness within the simulation community that model specification should be a distinct task quite separate from model implementation (simulation programming); see, for example, Nance <1984> and Oren, Zeigler and Elzas <1984>. This is in keeping with the structured modeling principles of having a rigorous theoretical modeling framework (feature a) and of separating model representation and model solution (feature b). Modeling frameworks for discrete event simulation have been constructed from general systems theory (Zeigler <1976> and Oren, Zeigler and Elzas <1984>) and from classical mathematics in the form of the entity-attribute-set (EAS) formalism of Markowitz and others (e.g., Markowitz <1979>), to name two of the more prominent approaches. Such frameworks provide the basis for model specification languages, which can be non-executable like CS (Overstreet and Nance <1985>) or executable like *SIMSCRIPT* (Markowitz <1979>). Unfortunately, it is still common practice to do simulation using FORTRAN or another general-purpose high level language in an ad hoc manner that confounds model with solver (Nance <1984>).

Another interesting development is the recognition in some quarters that model development and model management environments are needed that cater to as much of the simulation modeling life-cycle as possible (e.g., Nance <1981>). This is in keeping with structured modeling's life-cycle orientation (feature d). Nance <1981> proposes the so-called Conical Methodology as an approach to supporting many of the life-cycle phases. (That paper also advocates many other tenets held by structured modeling, including the need for a genuine modeling framework, the importance of complete documentation, the desirability of top-down design, and the need for an interactive implementation.)

One recent trend in model development systems is the rapid proliferation of implementations based on computer graphics. Examples include DRAFT (Mathewson <1985>), GSS (Zamanzadeh <1986>), the current version of SIMAN (Pegden <1985>), TESS (Standridge <1985>), and a rapidly growing family of animated simulation systems for what is often known as visual interactive modeling (Bell <1985>, Hurrion <1986>). This trend is interesting from a structured modeling viewpoint for at least two reasons. First, the model representations used in these systems tend to be declarative rather than procedural in character, and are graph-based rather than string-based. Thus they are closer to the structured modeling style of model representation than other kinds of simulation systems are. Second, most graphics-based systems are intended to reach a broader base of potential simulation users by achieving simpler user interfaces, and by exploiting the rapid and continuing improvement in graphics hardware performance/cost ratios. Structured modeling has similar aspirations. It is likely that there is much to learn from the evolution of simulation modeling systems based on computer graphics as they mature to meet practitioners' needs. The lessons should be particularly valuable if a graphics-oriented structured modeling implementation is undertaken centering on genus and element graphs.

The final simulation development to be mentioned is an evolutionary one in keeping with structured modeling's ambition to integrate database capabilities with modeling (feature g). Markowitz, Malhotra, and Pazel <1984> have taken the entity-attribute-set formalism together with its *SIMSCRIPT* implementation and extended these to a quite general application development system called EAS-E capable not only of modeling and database management, but of almost anything that can be done with a high level procedural language. We shall return to a discussion of EAS-E after first discussing the EAS formalism in somewhat greater detail.

The EAS formalism is closely related to certain structured modeling concepts. Its "entities" correspond to either primitive or compound entity elements in structured modeling terms. Its "attributes" correspond to either compound entity, attri-

bute, or function elements. Its "sets" correspond to compound entities that call their "owner" as well as their "members". Its "entity types" correspond to genera obeying a condition similar to generic similarity. The EAS formalism is used to describe the instantaneous "status" of the system being simulated. Separate procedural code specifies how system status changes over time.

Markowitz <1979> notes that the relational database formalism (without including functional dependencies) and the EAS formalism are "equally general". Since it is known that structured modeling subsumes the relational database formalism (even with functional dependencies), it appears to follow that structured modeling subsumes the EAS formalism. Of course, this subsumption is not necessarily a natural one. The best way to resolve the issue of natural subsumption would be to devise a straightforward translation procedure from an EAS model to a structured model.

Not only did Markowitz <1979> demonstrate that the EAS formalism used by *SIMSCRIPT* is as general as today's most dominant data model, but he also devotes considerable space to a discussion of the relation between *SIMSCRIPT* and several other data models. If this concern with the relationship between modeling languages and data models seems ahead of its time, consider that extensive database capabilities were planned (but not implemented) as part of *SIMSCRIPT II* Level 6 since the late 1960's.

The database capabilities and more were fulfilled with the EAS-E system developed and in use at IBM (Markowitz, Malhotra and Pazel <1984>). EAS-E is an application development system that includes "a procedural language for manipulating database and main-storage entities, and direct (nonprocedural) facilities for interrogating and updating database entities." EAS-E code is surprisingly compact by comparison with other high level languages, and is said to be fairly easy for novices to read (in another paper the authors call it "executable documentation"). Its browsing and updating facilities look quite friendly. A companion paper -- Markowitz, Malhotra and Pazel <1981> -- argues convincingly that a truly integrated application development system should use a single modeling framework not only for the modeling and database entities defined by the user, but also for all entities with which the user must interact, including graphical entities, text entities, and entities defined by the computer system itself. It could be worthwhile to consider structured modeling's potential as the basis for integrated application development systems of this type.

Now that discrete event simulation has been discussed from the point of view of structured modeling, it is natural to ask whether structured modeling can support discrete event simulation. This question is taken up in Appendix 2.

9. Graph Grammar-Based Systems

Rapid advances in computer graphics hardware and software guarantee the growing importance of modeling systems with graphically oriented model representations. One of the most difficult challenges facing the design of such systems is how to generate entire classes of models nearly as easily as one can generate a specific model instance. One promising answer to this challenge is provided by the notion of a "graph grammar".

Jones <1985> extends some recent work in graph grammars and demonstrates the applicability of these ideas to analytical modeling systems based on computer graphics. Only a partial implementation is available at the present time, but it is clear that this approach has considerable potential.

Jones is concerned with designing what might be called (but he does not) "modeling system generators". These dwell two levels of abstraction above specific model instances. The first level of abstraction, which he calls the "generic model", is a class of specific model instances all similar to one another in important respects. The second level of abstraction, which he calls the "meta-model", is a class of generic models with common mathematical structure. Jones chooses attributed, directed graphs as the mathematical structure at the second level.

Particular classes of attributed graphs can represent a variety of generic models; Jones cites network flow and vehicle routing models as examples. The crux of the matter is how tc characterize the common structure shared by all instances of a model class that a modeler wants to view as a generic model. Jones' solution to this puzzle is to adapt the theory of graph grammars to the task. This yields a generative mechanism for constructing any desired instance of a generic model. This mechanism can serve as the organizing principle of a computer-based implementation.

Jones' "meta-model" is a modeling framework in the same sense that structured modeling is. An implementation of it would be a modeling system (generator) in the same sense that a structured modeling implementation is. His "generic model" corresponds to the notion of a schema in structured modeling. Although he cites as illustrative generic models only examples that clearly involve graphs, we know from the wide scope of applicability of structured modeling and its attributed graph roots that Jones' work applies to a far broader range of generic models. This conclusion is reinforced by the fact that the conceptual graphs of Sowa <1984> can also be viewed as a modeling framework based on attributed graphs.

Jones' work is now appraised relative to the standard list of eight desirable features, after which its relation to structured modeling is discussed further.

Feature (a). The notion of an attributed graph as formalized in Jones <1985> provides a fully rigorous and coherent conceptual framework for modeling. The focal (generic) model representation is computer-executable in principle, but is in a grammar theoretic notation that most people would find obscure as a basis for either managerial communication or conventional mathematical use. For building or explaining a model instance, however, a system that can compile the grammar theoretic notation has excellent potential for good managerial communication through the power of graphic images.

Feature (b). Jones' approach has the potential for good independence between model representation and model solution.

Feature (c). As observed earlier, attributed graphs provide a modeling framework general enough to encompass a great many modeling paradigms.

Feature (d). Jones is primarily concerned with just part of the entire modeling life-cycle.

Feature (e). Representational independence of general structure and detailed data is a consequence of Jones' careful distinction between a generic model and an instance of a generic model.

Feature (f). There is good potential for friendly desktop implementation on the next generation of desktop computers and workstations, for they will be quite graphics-capable.

Feature (g). Jones does not consider the possibility of integrated database functions.

Feature (h). Immediate expression evaluation capability may be possible, but is left by Jones as a desirable extension.

There is potential mutual applicability between structured modeling and Jones' work. In one direction, structured modeling can provide an intermediate language through which a wide variety of model classes can be recast as generic attributed graphs ripe for the application of Jones' apparatus. In the other direction, Jones' apparatus could help (i) to identify useful standard transformations either of genus graphs or element graphs in structured modeling (cf Sowa <1984> on "formation rules"), and (ii) as the basis for a fully graphics-based implementation of structured modeling.

A particularly intriguing research issue is whether and how Jones' attributed graph grammar for a generic model can be translated automatically to/from a proper structured modeling schema. A "from" (resp. "to") translator could provide a "front end" (resp. "back end") facility for the kind of system Jones

envisions, and a "from" translator would enable Jones' system to be used as a graphical model-viewing and model-building medium for the kind of system envisioned in Geoffrion <1986a>.

Another research question is whether the "programmed graph grammars" Jones discusses, which allow a partial order to be established over the operations to be performed, could help to formalize the prompting tasks for which the smart loader/editor (see Geoffrion <1986a>) has responsibility.

10. Special Purpose Systems

The stronger the assumptions one makes concerning the class of models for which a modeling system will be used, the more effectively can the design of the system be tailored to the target applications.

There are numerous specialized modeling systems for engineering design (e.g., Westerberg <1985>), distribution system design (e.g., Geoffrion, Graves and Lee <1982>), manufacturing modeling (Engelke et al <1985>), project management (Assad and Wasil <1985>), vehicle routing and scheduling (Golden, Bodin and Goodwin <1985>), and so on (see, e.g., the Applications Software section of Data Sources <1986>).

All such systems are limited by definition in the generality of their conceptual frameworks, that is, they are weak by desirable feature (c). For this reason, they are not considered further here.

CONCLUSION

This concludes discussion of various modeling approaches and systems in the context of the eight desirable features advocated in Geoffrion <1986a>. A similar analysis for structured modeling itself can be found in Section 2.4 of that paper.

Several of the references in this paper have influenced the design of the prototype structured modeling implementation now under way at UCLA. At a more fundamental level, the theoretical literature from which many of the references spring -- especially database theory, discrete mathematics, and parts of computer science -- has influenced the structured modeling framework itself. As explained at various points in this paper, and in greater detail in Section 5 of Geoffrion <1986a>, there are many opportunities for additional influence.

We close by reiterating the point made earlier that this paper is not intended to criticize the alternative modeling approaches and systems reviewed here for failing to have all

eight desirable features, but rather to clarify how structured modeling differs from previous approaches and to point out some of the grounds for future cross-fertilization.

BIBLIOGRAPHY

ASSAD, A. and E. WASIL <1985>. "Project Management Using a Microcomputer," Working Paper MS/S 85-027, College of Business and Management, University of Maryland at College Park.

BARR, A. and E.A. FEIGENBAUM <1981>. *The Handbook of Artificial Intelligence*, Volume 1, William Kaufmann, Los Altos, CA.

BATORY, D.S. and M. MANNINO <1986>. "Panel on Extensible Database Systems," *ACM SIGMOD 86*.

BELL, P.C. <1985>. "Visual Interactive Modeling as an Operations Research Technique," *Interfaces*, 15:4 (July-August), 26-33.

BISSCHOP, J. and A. MEERAUS <1982>. "On the Development of a General Algebraic Modeling System in a Strategic Planning Environment," *Math. Programming Stud. 20* (October), North-Holland, Amsterdam, 1-29.

BODILY, S. <1986>. "Spreadsheet Modeling as a Stepping Stone," *Interfaces*, 16:5 (September-October), 34-52.

BRACHMAN, R.J. <1979>. "On the Epistemological Status of Semantic Networks," in N.V. Findler (ed), *Associative Networks: Representation and Use of Knowledge by Computers*, Academic Press, New York.

BRACHMAN, R.J. and H.J. LEVESQUE <1985>. *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, CA.

BURGER, W.F. <1981>. "A Modeling System for Mathematical Programming," Technical Report 177, Department of Computer Sciences, University of Texas, Austin, May.

BURGER, W.F. <1982>. "MLD: A Language and Data Base for Modeling," IBM Research Division, San Jose, Research Report RC 9639 (#42311), September 14.

CHILDS, C. and C.R. MEACHAM <1985>. "ANALYTICOL - An Analytical Computing Environment," *AT&T Technical J.*, 64:9 (November), 1995-2007.

CHILDS, C. and F.I. VOKOLOS <1986>. "AWB-ADE: An Application Development Environment for Interactive, Integrated Systems," working paper, AT&T Bell Laboratories, Warren, NJ 07060.

DATA SOURCES <1986>. Quarterly reference publication, Ziff-Davis Publishing Co., New York.

DERMAN, E. and E.G. SHEPPARD <1985>. "HEQS - A Hierarchical Equation Solver," *AT&T Technical J.*, 64:9 (November), 2061-2096.

DOLK, D.R. <1986>. "A Generalized Model Management System for Mathematical Programming," *ACM Trans. Math. Software*, 12:2 (June), 92-125.

ELLISON, E.F.D. and G. MITRA <1982>. "UIMP: User Interface for Mathematical Programming," *ACM Trans. Math. Software*, 8:3 (September), 229-255.

ENGELKE, H., J. GROTRIAN, C. SCHEUING, A. SCHMACKPFEFFER, W. SCHWARZ, B. SOLF and J. TOMANN <1985>. "Integrated Manufacturing Modeling System," *IBM J. Res. Develop.*, 29:4 (July), 343-355.

FOURER, R. <1983>. "Modeling Languages Versus Matrix Generators for Linear Programming," *ACM Trans. Math. Software*, 9:2 (June), 143-183.

FOURER, R., D.M. Gay and B.W. Kernighan <1987>. "AMPL: A Mathematical Programming Language," Computing Science Technical Report No. 133, AT&T Bell Laboratories, Murray Hill, NJ 07974, January.

GEOFFRION, A. <1986a>. "Introduction to Structured Modeling," Working Paper No. 338, Graduate School of Management, UCLA, June, revised February 1987. To appear in *Proceedings of the Conference on Integrated Modeling Systems* (held at the University of Texas, Austin, October 1986) and, without the section on implementation, in *Management Sci.*, May 1987.

GEOFFRION, A. <1986b>. "A Structured Modeling Representation of the GAMS Static Model of the Mexican Steel Industry," informal note, Graduate School of Management, UCLA, August 7.

GEOFFRION, A., G. GRAVES and L. LEE <1982>. "A Management Support System for Distribution Planning," *INFOR*, 20:4 (November), 287-314.

GOLDEN, B., L. BODIN and T. GOODWIN <1985>. "Microcomputer-Based Vehicle Routing and Scheduling Software," Working Paper MS/S 85-024, College of Business and Management, University of Maryland at College Park.

GREENBERG, H.J. <1983>. "A Functional Description of ANALYZE: A Computer-Assisted Analysis System for Linear Programming Models," *ACM Trans. Math. Software*, 9:1 (March), 18-56.

HURRION, R.D. <1986>. "Visual Interactive Modeling," *European J. Oper. Res.*, 23:3 (March), 281-287.

JONES, C.V. <1985>. *Graph-Based Models*, Ph.D. Thesis, Cornell University.

KATZ, S., L.J. RISMAN and M. RODEH <1980>. "A System for Constructing Linear Programming Models," *IBM Systems J.*, 19:4, 505-520.

KENDRICK, D.A. and R. KRISHNAN <1986>. "A Comparison of Structured Modeling and GAMS," to appear in *Proceedings of the Conference on Integrated Modeling Systems* (held at the University of Texas, Austin, October 1986).

KENDRICK, D.A. and A. MEERAUS <1985>. *GAMS: An Introduction*, draft, The World Bank, February. The Scientific Press, Palo Alto, to appear.

KRASNOFF, B. and J. DICKINSON <1986>. "Project Database II," *PC Magazine*, 5:12 (June 24), 106-227.

LUCAS, C. and G. MITRA <1985>. "CAMPS: Preliminary User Manual," Department of Mathematics and Statistics, Brunel University, Middlesex, U.K., July.

MARKOWITZ, H.M. <1979>. "SIMSCRIPT," in J. Belzer, A.G. Holzman and A. Kent (ed), *Encyclopedia of Computer Science and Technology*, Marcel Dekker, New York.

MARKOWITZ, H.M., A. MALHOTRA and D.P. PAZEL <1981>. "The ER and EAS Formalisms for System Modeling, and the EAS-E Language," RC 8802, IBM T.J. Watson Research Center, Yorktown Heights, NY, April 4.

MARKOWITZ, H.M., A. MALHOTRA and D.P. PAZEL <1984>. "The EAS-E Application Development System: Principles and Language Summary," *Comm. ACM*, 27:8 (August), 785-799.

MATHEWSON, S.C. <1985>. "Simulation Program Generators: Code and Animation on a P.C.," *J. Oper. Res. Society*, 36:7, 583-589.

MATURANA, S. <1987>. "Comparative Analysis of Mathematical Modeling Systems," informal note, Graduate School of Management, UCLA, February.

NANCE, R.E. <1981>. "Model Representation in Discrete Event Simulation: The Conical Methodology," Technical Report CS81003-R, Department of Computer Science, Virginia Polytechnic Institute and State University, March 15.

NANCE, R.E. <1984>. "Model Development Revisited," Working Paper, Computer Science Department, Virginia Polytechnic Institute and State University.

OREN, T.I., B.P. ZEIGLER and M.S. ELZAS <1984>. *Simulation and Model-Based Methodologies: An Integrative View*, NATO ASI Series, Springer-Verlag, Berlin.

OVERSTREET, C.M. and R.E. NANCE <1985>. "A Specification Language to Assist in Analysis of Discrete Event Simulation Models," *Comm. ACM*, 28:2 (February), 190-201.

PALMER, K. <1984>. *A Model Management Framework for Mathematical Programming*, Wiley, New York.

PEGDEN, C.D. <1985>. "Introduction to SIMAN," *Proc. 1985 Winter Simulation Conference*.

PLANE, D.R. <1986>. *Quantitative Tools for Decision Support Using IFPS*, Addison-Wesley, Reading, MA.

ROY, A., L. LASDON and J. LORDEMAN <1986>. "Extending Planning Languages to Include Optimization Capabilities," *Management Sci.*, 32:3 (March), 360-373.

SCHITTKOWSKI, K. <1985>. "EMP: A Software System Supporting the Numerical Solution of Mathematical Programming Problems," Working Paper, Institut fur Informatik, Universitat Stuttgart.

SOWA, J.F. <1984>. *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, MA.

SOWA, J.F. and E.C. WAY <1986>. "Implementing a Semantic Interpreter Using Conceptual Graphs," *IBM J Res. Develop.*, 30:1 (January), 57-69.

STANDRIDGE, C.R. <1985>. "Performing Simulation Projects with the Extended Simulation System," *Simulation* (December).

TSICHRITZIS, D.C. and F.H. LOCHOVSKY <1982>. *Data Models*, Prentice-Hall, Englewood Cliffs, NJ.

WAREN, A.D., M.S. HUNG and L.S. LASDON <1986>. "The Status of Nonlinear Programming Software: An Update," Working Paper, Dept. of Computer and Information Science, Cleveland State University, November.

WESTERBERG, A.W. <1985>. "Aids for Engineering System Model Formulation," Working Paper, Department of Chemical Engineering, University of Wisconsin, Madison.

ZAMANZADEH, B. <1986>. *A Graphical Simulation System for Manufacturing Systems*, Ph. D. Dissertation, Department of Engineering, UCLA.

ZEIGLER, B.P. <1976>. *Theory of Modeling and Simulation*, Wiley, New York.

COMMERCIAL SOFTWARE REFERENCES

DataEase. Software Solutions Inc., 12 Cambridge Dr., Trumbull, CT 06611.

dBASE III Plus. Ashton-Tate, 20101 Hamilton Ave., Torrance, CA 90502.

IFPS. Execucom Systems Corp., 3410 Far West Blvd., Austin, TX 78731.

IFPS/Personal. Execucom Systems Corp., 3410 Far West Blvd., Austin, TX 78731.

KnowledgeMan. Micro Data Base Systems Inc., P.O. Box 248, Lafayette, IN 47902.

LINDO. LINDO Systems Inc., P.O. Box 148231, Chicago, IL 60614.

MIMI/LP. Chesapeake Decision Sciences, Inc., 200 South Street, New Providence, NJ 07974.

MindSight. Execucom Systems Corp., 3410 Far West Blvd., Austin, TX 78731.

OPTIMUM. Execucom Systems Corp., 3410 Far West Blvd., Austin, TX 78731.

PAM. Ketron, Inc., 151 S. Warner Rd., Wayne, PA 19807.

Paradox. Ansa Software, 1301 Shoreway Rd. #221, Belmont, CA 94002.

R:base 5000. Microrim Inc., 3380 146th Pl. SE, Bellvue, WA 98007.

Revelation. COSMOS Inc., 19530 Pacific Hwy. South, Seattle, WA 98188.

SIMSCRIPT II.5. CACI, 3344 North Torrey Pines Court, La Jolla, CA 92037.

What's Best!. General Optimization Inc., 2251 North Geneva Terrace, Chicago, IL 60614.

1-2-3. Lotus Development Corporation, 161 First Street, Cambridge, MA 02142.

Appendix 1

ON THE RELATIONSHIP BETWEEN CONCEPTUAL GRAPHS AND STRUCTURED MODELING

The relationship between conceptual graphs (Sowa <1984>) and structured modeling's genus and element graphs is surprisingly intimate, and can be stated informally as follows:

1. Each concept node with a *referent* corresponds to an element or genus of primitive entity type. It corresponds to a singleton genus and an element if its referent is an *individual marker* (the marker is like an identifier in structured modeling, except that it must be a serial number). It corresponds to a singleton genus if its referent is a *generic marker*. It corresponds to a self-indexed genus if its referent is a *generic set*. And it corresponds to a self-indexed genus and one or more elements if its referent is a specific set.

2. Each concept node with no referent (assuming no implicit generic marker) corresponds to an element or singleton genus of compound entity type. A concept node may also take as its referent an entire set of conceptual graphs, in which case it can be viewed as a singleton compound entity.

3. As indicated earlier, nearly all conceptual relation nodes have degree two and so necessarily link two concept nodes. Each such conceptual relation node corresponds to an arc in a genus or element graph. Alternatively, each such node can be viewed as spelling out explicitly the role played by a specific call in a specific calling sequence.

4. Conceptual graphs do not incorporate the attribute type of structured modeling directly. Sowa deals with a text-valued attribute as a *word* (a particular type of concept) that is associated with an *entity* (also a type of concept) through a *name* (a type of conceptual relation). He deals with a numerical-valued attribute as a word that is associated with a *measure* (a type of concept) through a particular type of conceptual relation which is also called *measure*. This corresponds to a standard trick in structured modeling of replacing an attribute element by a compound entity element with the help of a primitive entity element that represents the particular value. Sowa does recognize that this approach can be cumbersome, and so introduces *name contraction* and *measure contraction* conventions to simplify conceptual graphs where "name" and "measure" are used. The resulting simplified graphs contain contracted concept nodes that are similar to singleton attribute genera together with their elemental detail.

5. Conceptual graphs do not have anything that corresponds to the function or test elements of structured modeling, but provision is made for connecting conceptual graphs to *functional data flow graphs* that do embody constructs that act like such elements.

It follows from points 1, 2, and 3 that any conceptual graph with all conceptual relation nodes of degree two corresponds to a genus graph with only primitive and compound entity nodes. This can be demonstrated constructively by erasing each conceptual relation node and merging its incoming and outgoing arcs, possibly changing the orientation of the resulting arc. Any aspect of the conceptual graph not directly translated to the genus graph can be viewed as elemental detail.

A partial reverse correspondence appears to be possible for structured models with no attributes, functions, or tests. The principal change is the introduction of conceptual relation nodes to explain the role played by each call (this explanation is supposed to appear in the interpretation part of each genus paragraph).

Here are some additional observations on the connections between Sowa's development and structured modeling.

6. There is no provision for explicit indexing in conceptual graphs.

7. Sowa makes a strong distinction between *intension* (roughly, the meaning of something in an abstract sense) and *extension* (roughly, instances of something in a concrete sense). His *concepts* and *conceptual relations* are intensional, while his referents are extensional. The corresponding distinction in structured modeling is captured by the separation of generic structure from elemental structure. There appears to be nothing in Sowa's development that corresponds to modular structure, although he uses a related idea as explained in the next comment.

8. The concepts appearing in conceptual graphs are understood not in an isolated sense, but rather as occurring within a broader context. That context is provided by a *type hierarchy*: a partial ordering of concept categories. The type hierarchy is an adjunct to a conceptual graph and is not always needed. It is particularly helpful as a basis for property inheritance. Type hierarchies can be modeled within the structured modeling framework, but it is not common practice to do so in connection with ordinary applications in MS/OR.

9. Sowa rigorously develops *formation rules* for editing and combining conceptual graphs. They constitute a context-free graph grammar. Viewed in reverse, they amount to rules for

inference. Although the formation rules all have analogs in structured modeling, no corresponding theory has yet been worked out. This is an inviting research topic.

10. Sowa comments extensively on the limitations of conceptual graphs. Some, such as the unrealism of the finiteness assumption in certain situations, are applicable also to structured modeling. Other limitations are not pertinent because structured modeling does not aspire to model the full range of natural language.

It is evident that Sowa's grand synthesis of the foundations of AI contains much material worth adapting to structured modeling. For example, it points the way to a better understanding of the relationship between structured modeling and both natural language and first-order logic.

The strong connection that Sowa develops between first-order logic and conceptual graphs (and hence structured modeling) suggests adding inference engines to the kinds of solvers that structured modeling seeks to accommodate. How to do this constitutes a potentially important research topic.

Comment on the Concept of a "Solver"

This last point suggests that the very concept of a "solver" should be rethought. This term is based on the notion of a "solution", which reflects a analytical modeling orientation in which the main concern is with solving systems of equations and optimization. The term "retrieval" would better reflect the orientation of the database community, which is strongly concerned with retrieving data elements and rearranging them so as to answer queries posed against a database. The term "inference" would better reflect the orientation of the artificial intelligence community, one of whose principal concerns is the drawing of inferences from a knowledge base. The last of these terms is probably the most appropriate if a model-based synthesis of all three disciplines is to be achieved. The solution to a set of equations or to an optimization problem is certainly a kind of inference based on model assumptions, and the answer to a query requires a kind of inference based on a database.

Let us, then, think of a "solver" as a device for drawing a class of inferences (using the term broadly) from a model. The inferences can be mathematically elementary as in the case of database systems, mathematically advanced as is often the case in MS/OR systems, or somewhere in between as usually is the case in artificial intelligence.

Appendix 2

HOW CAN DISCRETE EVENT SIMULATION BE DONE WITHIN THE STRUCTURED MODELING FRAMEWORK?

One possible answer to the question of the title of this appendix is to prepare a static structured model of the system to be simulated and to compose a (probably procedural) control program that edits elemental structure according to the rules governing the system's dynamic behavior. This could involve creating and deleting elements, changing attribute values, and so on. The control program would also determine when the evaluation operation should be carried out, record the pertinent aspects of the dynamic behavior of the elemental structure and, after execution, report a summary of this behavior and restore the elemental structure to its original state (unless only a copy of it was manipulated).

It would, of course, be objectionable for the control program to contain information more properly thought of as part of the specification of the target system's dynamics. Such information should be included as part of the structured model itself, in a dynamic part of the schema to complement the static part mentioned above. The control program should be designed to use the dynamics described in a structured model to govern the editing of elemental structure, and also to accept directions from the user as to the nature of the simulation experiment to be performed. These directions, communicated by the user in a simple language rather than by writing procedural code, should have to do only with the user's intentions and not with the model per se. They can be thought of as specifying a task to a solver in connection with a specific model.

Thus the control program would in effect become a kind of all-purpose solver for discrete event simulation in the context of structured modeling. It would not need to be customized for each application. No such solver has yet been built, and it is not obvious whether the idea is practical. It is encouraging to observe that *SIMSCRIPT* can be viewed as working according to a similar plan (except that customization is required for each application). Its EAS formalism corresponds to the part of the structured model that describes the system's static aspects, and its event routines correspond to the part of the structured model that describes the system's dynamics together with the user's directions concerning the experiment to be performed.

The development of these ideas for achieving discrete event simulation capability within the structured modeling framework is an open research opportunity at this time.

END

5-81

DTIC